DSC 440, HW5

Kefu Zhu

10.2

Suppose that the data mining task is to cluster points (with (x, y) representing location) into three clusters, where the points are

$$A_1(2,10), A_2(2,5), A_3(8,4), B_1(5,8), B_2(7,5), B_3(6,4), C_1(1,2), C_2(4,9)$$

The distance function is Euclidean distance. Suppose initially we assgin A_1, B_1, and C_1 as the center of each cluster, respectively. Use the $k-means$ algorithm to show *only*

```
# Import library
import numpy as np
from sklearn.cluster import KMeans
seed = 0
```

(a) The three cluster centers after the first round of execution

Answer:

```
data = np.array([[2,10],[2,5],[8,4],[5,8],[7,5],[6,4],[1,2],[4,9]])
fix_init = np.array([[2,10],[5,8],[1,2]])
my_kmeans = KMeans(n_clusters=3, init=fix_init, random_state=seed, max_iter=1).fit
```

The cluster centers after the first round of execution are `(2,10)`, `(6,6)` and `(1.5,3.5)`

(b) The final three clusters

Answer:

```
my_kmeans = KMeans(n_clusters=3, init=fix_init, random_state=seed).fit(data)
my_kmeans.predict(data)
```

The final three clusters are

- Cluster 1: $A_1(2, 10), B_1(5, 8), C_2(4, 9)$
- Cluster 2: $A_3(8, 4), B_2(7, 5), B_3(6, 4)$
- Cluster 3: $A_2(2, 5), C_1(1, 2)$

10.4

For the $k - means$ algorithm, it is interesting to note that by choosing the initial cluster centers carefully, we may be able to not only speed up the algorithm's convergence, but also guarantee the quality of the final clustering. The $k - means + +$ algorithm is a variant of $k - means$, which chooses the initial centers as follows.

First, it selects one center uniformly at random from the objects in the data set. Iteratively, for each object p other than the chosen center, it chooses an object as the new center. This object is chosen at random with probability proportional to $dist(p)^2$, where $dist(p)$ is the distance from p to the closest center that has already been chosen. The iteration continues until k centers are selected.

Explain why this method will not only speed up the convergence of the $k - means$ algorithm, but also guarantee the quality of the final clustering results.

Answer:

By using $k - means + +$, we will end up with initial cluster centers that are far apart from each other, because data points that are far away from the closest center that has already been chosen will have higher value for $dist(p)$, hence have higher chance of being selected as the next center.

As a result, we will be able to avoid initializations where the initial clusters are really close to each other, which will makes the computation time needed to reach convergence much longer.

10.6

Both $k - means$ and $k - medoids$ algorithms can perform effective clustering.

(a) Illustrate the strength and weakness of $k - means$ in comparison with $k - medoids$.

Answer:

$k - means$ suffers from issue in the data such as outliers, which will have higher influence on deciding the center of the clusters because we compute the average distance. However, by using $k - medoids$, intead of taking the mean value of the data points in a cluster as the proposing new cluster center, we choose the most centrally located data point (medoid). Hence, $k - medoids$ is more robust to outliers and extreme values in the data.

On the other hand, $k-means$ is faster than $k-medoid$ because it takes less time when computing the new centroid for the cluster in each iteration.

(b) Illustrate the strength and weakness of these schemes in comparison with a hierarchical clustering scheme (e.g., AGNES).

Answer:

Both $k-means$ and $k-medoids$ are partition clustering methods, which work well for small to medium sized dataset, and are also capable of undoing previous partition decision when needed. But partition clustering methods need to specify the number of clusters to partition beforehand, which is often an unknown information and require parameter testing.

In hierarchical clustering, user does not need to specify the number of clusters when executing the algorithm. Instead, the user can look at the result and choose to merge to certain degree that the clustering result can be representative enough for the data situation. The disadvantage is that in hierarchicaal clustering, you cannot undo the merging steps.

9.1

The following table consists of training data from an employee database. The data have been generalized. For example, "$31...35$" for age represents the age range of 31 to 35. For a given row entry, count represents the number of data tuples having the values for department, status, age, and salary given in that row.

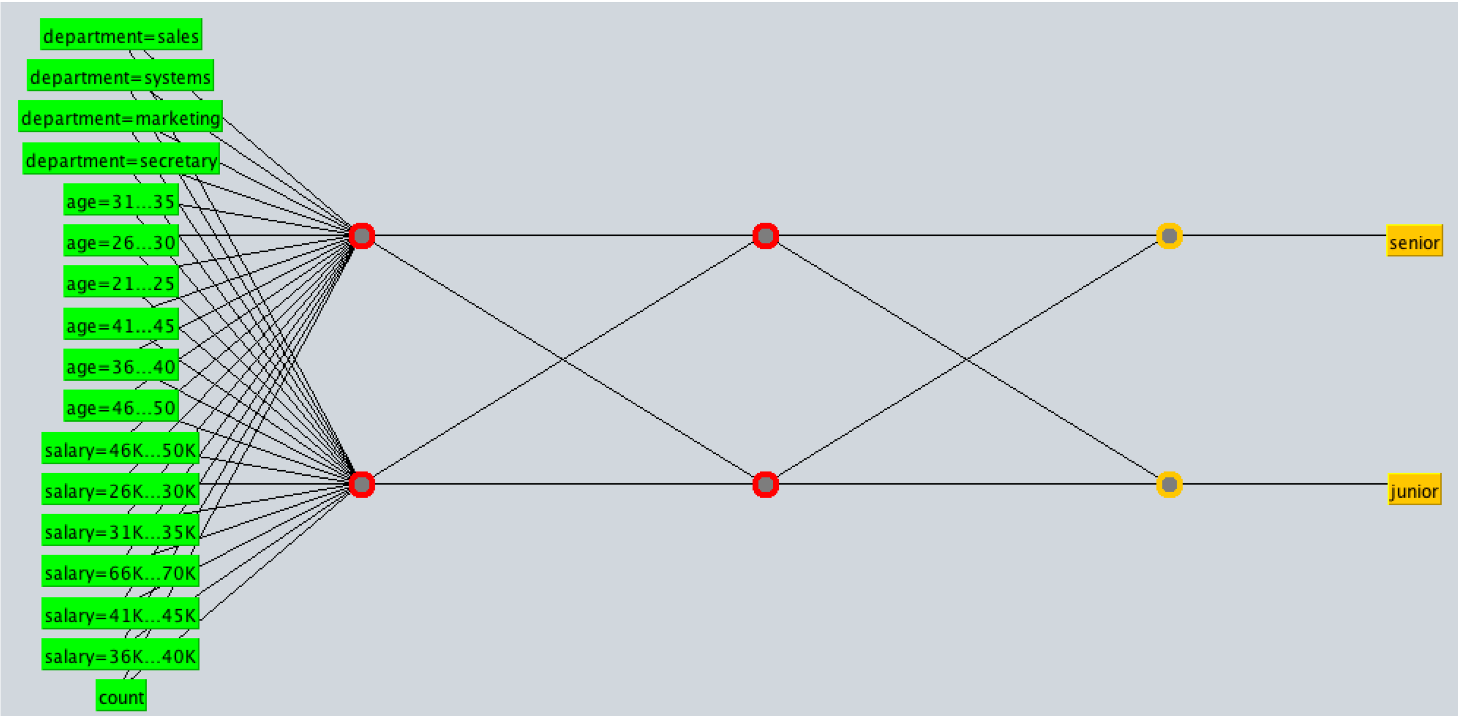
department	*status*	*age*	*salary*	*count*
sales	senior	31 … 35	46K … 50K	30
sales	junior	26 … 30	26K … 30K	40
sales	junior	31 … 35	31K … 35K	40
systems	junior	21 … 25	46K … 50K	20
systems	senior	31 … 35	66K … 70K	5
systems	junior	26 … 30	46K … 50K	3
systems	senior	41 … 45	66K … 70K	3
marketing	senior	36 … 40	46K … 50K	10
marketing	junior	31 … 35	41K … 45K	4
secretary	senior	46 … 50	36K … 40K	4
secretary	junior	26 … 30	26K … 30K	6

Let status be the class-label attribute

1. Multiplayer Neural Network

(a) Design a multilayer feed-forward neural network for the given data. Label the nodes in the input and output layers.

To make the weight report shorter for the next question, I only build a 2-hidden-layer neural network with two nodes in each layer. The network structure looks like this



(b) Using the multilayer feed-forward neural network obtained in (a), show the weight values after one iteration of the backpropagation algorithm, given the training instance “(sales, senior, 31 . . . 35, 46K . . . 50K)”. Indicate your initial weight values and biases and the learning rate used.

Initial weight values are listed below:

```
Sigmoid Node 0
    Inputs    Weights
    Threshold    -0.04782492348275749
    Node 4    0.005556492623585749
    Node 5    0.013255808378373977
Sigmoid Node 1
    Inputs    Weights
    Threshold    0.01896096065774131
    Node 4    -0.013518839455665444
    Node 5    0.05924433310607999
Sigmoid Node 2
    Inputs    Weights
    Threshold    0.04853427033674249
    Attrib department=sales    0.04639244145814951
    Attrib department=systems    0.012469314924797645
    Attrib department=marketing    0.02763931692278499
    Attrib department=secretary    -0.001283469694506863
    Attrib age=31…35    0.02331020671184633
    Attrib age=26…30    0.0337543072231283
```

```
    Attrib age=21…25    0.039882833413158854
    Attrib age=41...45    -0.041730516774105594
    Attrib age=36…40    0.02225629052556144
    Attrib age=46…50    -0.03567185290984911
    Attrib salary=46K…50K    -0.049554069739043166
    Attrib salary=26K…30K    -0.015249614021838465
    Attrib salary=31K…35K    0.03588763460477099
    Attrib salary=66K…70K    0.03657290881144306
    Attrib salary=41K…45K    -0.0321638086752647
    Attrib salary=36K…40K    0.03545482526432028
    Attrib count    0.01918268996741153
Sigmoid Node 3
    Inputs    Weights
    Threshold    0.03794543897280069
    Attrib department=sales    -0.03956932422620263
    Attrib department=systems    -0.008946692228383638
    Attrib department=marketing    0.04907020560307811
    Attrib department=secretary    0.024616543758317507
    Attrib age=31…35    0.03172500991655066
    Attrib age=26…30    0.0026316265744914754
    Attrib age=21…25    -0.036623188301218
    Attrib age=41...45    0.04782599251552038
    Attrib age=36…40    0.021497500662934352
    Attrib age=46…50    -0.003715738208307521
    Attrib salary=46K…50K    -0.0428554290490922
    Attrib salary=26K…30K    -0.01615025480150633
    Attrib salary=31K…35K    0.04713884018973529
    Attrib salary=66K…70K    0.011232880598765002
    Attrib salary=41K…45K    -0.02826661821087119
    Attrib salary=36K…40K    -0.04904417032045238
    Attrib count    0.027122377396842694
Sigmoid Node 4
    Inputs    Weights
    Threshold    0.04083304025416515
    Node 2    -0.024111078095310264
    Node 3    -0.038797461052745556
Sigmoid Node 5
    Inputs    Weights
    Threshold    -0.03802364374401687
    Node 2    -0.04894292789271556
    Node 3    0.0032603429362751206
```

The learning rate is 0.3, and the weight after first iteration on the first data sample is

```
Sigmoid Node 0
    Inputs    Weights
    Threshold    -0.04782492348275749
    Node 4    0.005556492623585749
    Node 5    0.013255808378373977
Sigmoid Node 1
    Inputs    Weights
    Threshold    0.01896096065774131
    Node 4    -0.013518839455665444
    Node 5    0.05924433310607999
Sigmoid Node 2
    Inputs    Weights
    Threshold    0.04853427033674249
    Attrib department=sales    0.04639244145814951
    Attrib department=systems    0.012469314924797645
    Attrib department=marketing    0.02763931692278499
    Attrib department=secretary    -0.001283469694506863
    Attrib age=31…35    0.02331020671184633
    Attrib age=26…30    0.0337543072231283
    Attrib age=21…25    0.039882833413158854
    Attrib age=41...45    -0.041730516774105594
    Attrib age=36…40    0.02225629052556144
    Attrib age=46…50    -0.03567185290984911
    Attrib salary=46K…50K    -0.049554069739043166
    Attrib salary=26K…30K    -0.015249614021838465
    Attrib salary=31K…35K    0.03588763460477099
    Attrib salary=66K…70K    0.03657290881144306
    Attrib salary=41K…45K    -0.0321638086752647
    Attrib salary=36K…40K    0.03545482526432028
    Attrib count    0.01918268996741153
Sigmoid Node 3
    Inputs    Weights
    Threshold    0.03794543897280069
    Attrib department=sales    -0.03956932422620263
    Attrib department=systems    -0.008946692228383638
    Attrib department=marketing    0.04907020560307811
    Attrib department=secretary    0.024616543758317507
    Attrib age=31…35    0.03172500991655066
    Attrib age=26…30    0.0026316265744914754
    Attrib age=21…25    -0.036623188301218
    Attrib age=41...45    0.04782599251552038
    Attrib age=36…40    0.021497500662934352
    Attrib age=46…50    -0.0037157382083075 21
    Attrib salary=46K…50K    -0.0428554290490922
```

```
    Attrib salary=26K…30K    -0.01615025480150633
    Attrib salary=31K…35K    0.04713884018973529
    Attrib salary=66K…70K    0.011232880598765002
    Attrib salary=41K…45K    -0.02826661821087119
    Attrib salary=36K…40K    -0.04904417032045238
    Attrib count    0.027122377396842694
Sigmoid Node 4
    Inputs    Weights
    Threshold    0.04083304025416515
    Node 2    -0.023111078095310264
    Node 3    -0.039797461052745556
Sigmoid Node 5
    Inputs    Weights
    Threshold    -0.03802364374401687
    Node 2    -0.04694292789271556
    Node 3    0.0034603429362751206
```

2. SVM

The coefficients of SVM model after training looks like this

```
=== Classifier model (full training set) ===

SMO

Kernel used:
  Linear Kernel: K(x,y) = <x,y>

Classifier for classes: senior, junior

BinarySMO

Machine linear: showing attribute weights, not support vectors.

        -0.1307 * (normalized)  department=sales
 +       0.3128 * (normalized)  department=systems
 +       0.0729 * (normalized)  department=marketing
 +      -0.255  * (normalized)  department=secretary
 +      -0.1664 * (normalized) age=31…35
 +       0.9261 * (normalized) age=26…30
 +       0.7862 * (normalized) age=21…25
 +      -0.2169 * (normalized) age=41...45
 +      -0.8913 * (normalized) age=36…40
 +      -0.4376 * (normalized) age=46…50
 +      -0.3616 * (normalized) salary=46K…50K
 +       0.1826 * (normalized) salary=26K…30K
 +       0.8693 * (normalized) salary=31K…35K
 +      -1.2169 * (normalized) salary=66K…70K
 +       0.9643 * (normalized) salary=41K…45K
 +      -0.4376 * (normalized) salary=36K…40K
 +       0.3071 * (normalized) count
 +       0.1217

Number of kernel evaluations: 61 (92.269% cached)
```

The model performance is shown as below

```
=== Evaluation on training set ===

Time taken to test model on training data: 0 seconds

=== Summary ===

Correctly Classified Instances          11               100      %
Incorrectly Classified Instances         0                 0      %
Kappa statistic                          1
Mean absolute error                      0
Root mean squared error                  0
Relative absolute error                  0      %
Root relative squared error              0      %
Total Number of Instances               11

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000
Weighted Avg.    1.000    0.000    1.000      1.000    1.000      1.000    1.000

=== Confusion Matrix ===

 a b   <-- classified as
 5 0 | a = senior
 0 6 | b = junior
```

3. Logistic Regression

The coefficients of Logistic Regression after training looks like this

```
=== Classifier model (full training set) ===

Logistic Regression with ridge parameter of 1.0E-8
Coefficients...
                                        Class
Variable                               senior
=============================================
 department=sales                     15.4121
 department=systems                  -11.7066
 department=marketing                 -7.3458
 department=secretary                  5.0068
age=31…35                             11.1206
age=26…30                            -19.7511
age=21…25                            -31.6926
age=41...45                           14.3942
age=36…40                             19.1757
age=46…50                             14.3878
salary=46K…50K                        10.7576
salary=26K…30K                        -5.9401
salary=31K…35K                       -37.0433
salary=66K…70K                        19.7914
salary=41K…45K                       -32.3982
salary=36K…40K                        14.3878
count                                  0.0727
Intercept                             -6.1169
```

The model performance is shown as below

```
=== Evaluation on training set ===

Time taken to test model on training data: 0 seconds

=== Summary ===

Correctly Classified Instances          11              100      %
Incorrectly Classified Instances         0                0      %
Kappa statistic                          1
Mean absolute error                      0
Root mean squared error                  0
Relative absolute error                  0.0001 %
Root relative squared error              0.0001 %
Total Number of Instances               11

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000
Weighted Avg.    1.000    0.000    1.000      1.000    1.000      1.000    1.000

=== Confusion Matrix ===

 a b   <-- classified as
 5 0 | a = senior
 0 6 | b = junior
```

11.2

AllElectronics carries 1000 products, $P_1, \ldots, P_{1000}$. Consider customers Ada, Bob, and Cathy such that Ada and Bob purchase three products in common, P_1 ,P_2 , and P_3 . For the other 997 products, Ada and Bob independently purchase seven of them randomly. Cathy purchases 10 products, randomly selected from the 1000 products.

(1) In Euclidean distance, what is the probability that $dist(Ada, Bob) > dist(Ada, Cathy)$?

Answer:

Suppose Ada and Bob have x products purchased in common, the probability that happened can be computed as the following for different x:

$$P(x) = \frac{\binom{997}{7} \times \binom{7}{x-3} \times \binom{990}{10-x}}{\binom{997}{7}^2} = \frac{\binom{7}{x-3} \times \binom{990}{10-x}}{\binom{997}{7}}$$

Similarly, for Ada and Cathy, the probability can be computed as

$$P(x) = \frac{\binom{997}{7} \times \binom{10}{x} \times \binom{990}{10-x}}{\binom{997}{7}\binom{1000}{10}} = \frac{\binom{10}{x} \times \binom{990}{10-x}}{\binom{1000}{10}}$$

The formular to compute Jaccard similarity for both $Jaccard(Ada, Bob)$ and $Jaccaard(Ada, Cathy)$ is the same, given x purchases are the same:

$$Jaccard(Ada, Bob) = Jaccard(Ada, Cathy) = \frac{x}{20-x}$$

Use a little help from Python

```
from scipy.special import comb
import pandas as pd
import numpy as np

def Euclidean_Ada_Bob(x):
    numerator = comb(N=7,k=x-3)*comb(N=990,k=10-x)
    denominator = comb(N=997,k=7)
    return(numerator/denominator)

def Euclidean_Ada_Cathy(x):
    numerator = comb(N=10,k=x)*comb(N=990,k=10-x)
    denominator = comb(N=1000,k=10)
    return(numerator/denominator)
```

For $dist(Ada, Bob)$, we have

```
Ada_Bob = pd.DataFrame({'x':[x for x in range(3,11)],
                        'Euclidean_Distance':[np.sqrt(x) for x in range(3,11)],
                        'Jaccard_Distance':[x/(20-x) for x in range(3,11)],
                        'Probablity':[Euclidean_Ada_Bob(x) for x in range(3,11)]})
Ada_Bob
```

x	Euclidean_Distance	Jaccard_Distance	Probablity
3	1.732051	0.176471	9.517334e-01
4	2.000000	0.250000	4.739323e-02
5	2.236068	0.333333	8.660691e-04
6	2.449490	0.428571	7.319719e-06
7	2.645751	0.538462	2.966451e-08
8	2.828427	0.666667	5.404466e-11
9	3.000000	0.818182	3.643051e-14
10	3.162278	1.000000	5.256928e-18

Similarly, for $dist(Ada, Cathy)$, we have

```
Ada_Cathy = pd.DataFrame({'x':[x for x in range(1,11)],
                          'Euclidean_Distance':[np.sqrt(x) for x in range(1,11)],
                          'Jaccard_Distance':[x/(20-x) for x in range(1,11)],
                          'Probablity':[Euclidean_Ada_Cathy(x) for x in range(1,11
Ada_Cathy
```

x	Euclidean_Distance	Jaccard_Distance	Probablity
1	1.000000	0.052632	9.214765e-02
2	1.414214	0.111111	3.800387e-03
3	1.732051	0.176471	8.247703e-05
4	2.000000	0.250000	1.026772e-06
5	2.236068	0.333333	7.505338e-09
6	2.449490	0.428571	3.171627e-11
7	2.645751	0.538462	7.344917e-14
8	2.828427	0.666667	8.363392e-17
9	3.000000	0.818182	3.758406e-20
10	3.162278	1.000000	3.796369e-24

The probability of $dist(Ada, Bob) > dist(Ada, Cathy)$ can be computed as

$$\sum_{x=3}^{9}(P_{AdaBob}(x)\sum_{y=x+1}^{10} P_{AdaCathy}(y))$$

```
prob = 0
for x in range(3,10):
    AddCathy_probsum = 0
    for y in range(x+1,11):
        AddCathy_probsum += np.unwrap(Ada_Cathy.Probablity[Ada_Cathy.x == y])
    prob += np.unwrap(Ada_Bob.Probablity[Ada_Bob.x == x])*AddCathy_probsum
prob
```

$$\sum_{x=3}^{9}(P_{AdaBob}(x)\sum_{y=x+1}^{10} P_{AdaCathy}(y)) \approx 9.85 \times 10^{-7}$$

(2) What if Jaccard similarity (Chapter 2) is used?

Answer:

Similar to part 1

```
prob = 0
for x in range(3,10):
    AddCathy_probsum = 0
    for y in range(x+1,11):
        AddCathy_probsum += np.unwrap(Ada_Cathy.Probablity[Ada_Cathy.x == y])
    prob += np.unwrap(Ada_Bob.Probablity[Ada_Bob.x == x])*AddCathy_probsum
prob
```

For Jaccard similarity, the probability of $dist(Ada, Bob) > dist(Ada, Cathy)$ is computed to be

$$\sum_{x=3}^{10} (P_{AdaBob}(x) \sum_{y=1}^{x-1} P_{AdaCathy}(y)) \approx 0.096$$

(3) What can you learn from this example?

Answer:

- The larger the Euclidean distance is, the more dissimilar two objects are. But if measured in Jaccard similarity, the smaller the value is, the more dissimilar two objects are.